ASI Adaptive Sensors, Incorporated
[illegible] Suite 8 Santa Monica, CA 90405 (213) 396-5997

AN ALGEBRAIC SYNDROME DECODING TECHNIQUE FOR CERTAIN CONVOLUTIONAL CODES

I. S. Reed

25 October 1985

Combined Third and Final Progress Report

Submitted to

The Office of Naval Research

Arlington VA 22219

Under Contract #N00014-84-C-0720

by

ADAPTIVE SENSORS, INCORPORATED

AN ALGEBRAIC SYNDROME DECODING TECHNIQUE FOR CERTAIN CONVOLUTIONAL CODES

I. S. Reed

I. INTRODUCTION

In the first quarterly report [1], further progress was reported on syndrome decoding by extending the concept of error-trellis decoding to certain non-systematic convolutional codes (CCs). Syndrome decoding makes use of the fact that all CCs are capable of correcting only a finite number, say t, of errors in some small multiple of the constraint length. Also, using the syndrome either explicitly or implicitly, an error trellis is shown to exist for all systematic and certain non-systematic CCs and is independent of the original transmitted code. In fact, the states of an error trellis depend only on the channel errors. If there are no channel errors, the error trellis is in and remains in the zero state of the trellis.

In [1], it was shown for systematic and certain non-systematic CCs, including the non-systematic dual-k codes for which an error trellis can be defined, that the finite error-correcting capability of these CCs makes possible a reduction of sometimes a substantial number of states and transitions in the error trellis, compared with a coding trellis. This reduction in the number of states and paths makes error-trellis decoding potentially somewhat simpler than Viterbi or sequential decoding which utilizes a full coding trellis or tree. In [1], this reduction of complexity was demonstrated in an example wherein error-trellis decoding was applied to

the special case of a one error-correcting per blocklength, one-half rate, dual-k CC.

In this final report, some of the same machinery developed earlier for syndrome decoding is shown to provide a basis for what is called algebraic syndrome decoding of convolutional codes. Actually, algebraic syndrome decoders were developed quite early in a heuristic manner for certain simple convolutional codes. In particular, syndrome decoders in the form of feed-back decoders were developed for the one error-correcting Wyner-Ash convolutional code [e.g., Ref. 2 and other limited error-correcting CCs]. However, the generalization to algebraically constructed syndrome decoders other than feed-back and majority-logic decoders seems to have been largely ignored. In order to demonstrate the general nature of algebraic syndrome decoding, some definitions and terminology are now in order.

Suppose the information or message sequence, the input to the CC, is represented by

$$\underline{x}(D) = \left[x_1(D), \ldots, x_K(D)\right] , \tag{1a}$$

where

$$x_i(D) = \sum_{j=0}^{\infty} x_{ji} D^j \tag{1b}$$

for $1 \leq j \leq k$ are elements in $F[D]$, the ring of polynomials in the unit delay operator D over $F = GF(q)$, a Galois field, with q a power of a prime integer. Vector $\underline{x}(D)$ is a generating function in D of the input message sequence $\underline{x} = \left[\underline{x}_0, \ldots, \underline{x}_j, \ldots\right]$, where $\underline{x}_j = \left[x_{j1}, \ldots, x_{jk}\right]$ is a vector belonging to $V_k(F)$, the k-dimensional vector space over F. $\underline{x}(D)$ is sometimes

called a D-transform of the message or information sequence $\underline{x}$. The k component vector $\underline{x}_j$ in $\underline{x}$ is called the information frame at stage or frame time j.

Likewise, let the output sequence be

$$\underline{y}(D) = \left[y_1(D), \ldots, y_n(D) \right] , \tag{2}$$

where $y_i(D) \in F[D]$. Vector $\underline{y}(D)$ is the D-transform of output coded sequence $\underline{y} = \left[\underline{y}_0, \ldots, \underline{y}_j, \ldots \right]$, where $\underline{y}_j = \left[y_{j1}, \ldots, y_{jk} \right]$ belongs to $V_n(F)$. The n-vector $\underline{y}_j$ is called the j-th codeword frame of code sequence $\underline{y}$.

The information and code sequences of an (n, k) convolutional code are linearly related by a k × n, rank k, generator matrix G(D) of polynomial elements in F[D], as follows:

$$\underline{y}(D) = \underline{x}(D)\, G(D) . \tag{3}$$

The maximum degree m of the polynomial elements of G(D) in D is called the memory, and the constraint length L is defined as L = m + 1. Hence, matrix G(D) in Eq. (3) can be expressed as the finite D-transform or polynomial in D of form

$$G(D) = \sum_{j=0}^{m} G_j\, D^j , \tag{4}$$

where G_j are k × m matrices of elements in F[D].

Now, multiply Eq. (1a) by Eq. (4) on the right side of Eq. (3) and equate coefficients of D^j on both sides of Eq. (3). This yields the identity

$$\underline{y}_j = \sum_{i=0}^{\min (j,\, m)} \underline{x}_{j-i}\, G_i , \tag{5}$$

which is the convolution of sequence $\{\underline{x}_0, \underline{x}_1, \ldots\}$ of information frames with the sequence $\{G_0, G_1, \ldots, G_m\}$ of matrix operators in Eq. (4). The memory m of the convolution, Eq. (5), is the maximum number of past input frames, $\underline{x}_j$, needed to compute Eq. (5) recursively.

By Eq. (5), the j-th output n-vector of codeword frame $\underline{y}_j$ is dependent, at most, on the $m + 1 = L$ present and past input k-vectors of information frames. Hence, it is natural, as suggested by Blahut [2, Sec. 12.1], to define

$$k_1 = (m + 1)k = Lk$$

to be the wordlength of a convolutional code. Then, the wordlength k_1 is extended by the encoding process, Eq. (5), to what is called the blocklength, n_1, of the CC. The blocklength of a CC is

$$n_1 = (m + 1)n = Ln = k_1 R ,$$

where $R = k/n$ is the rate of the code. By Eq. (5), the blocklength $n_1 = (m + 1)n$ is the length of a subsequence of $\underline{y}$, which, during encoding, can be influenced by a single information frame.

Minimum distance between codeword segments of ℓ codeword frames is defined, usually, for only those pairs of codeword segments which differ in the first or initial frame:

Definition 1 (see [2, Sec. 12.3]). The ℓ-th minimum distance, d_ℓ, of a CC is the smallest Hamming distance between any two initial codeword segments of ℓ frames which differ or disagree in the initial frame. If $\ell = L = m + 1$, d_{m+1} is defined to be d, i.e., $d = d_{m+1}$. d is called the minimum distance of the code.

Since a CC is linear, one of the two codewords in Def. 1 can be chosen

to be the all-zero word. In this case, d_ℓ can be interpreted to be the Hamming weight of the smallest-weight codeword segment of ℓ frames, which is non-zero in the first frame. Thus, d_ℓ can be computed directly from a labeled coding trellis of the CC.

Suppose now, for some CC, that, at most, t errors occur during transmission in the first ℓ codeword frames, and that

$$2t + 1 \leq d_\ell$$

is satisfied by the code. Then those errors which occur in the first frame can be corrected. If $\ell = m + 1 = L$, then t satisfies

$$2t + 1 \leq d .$$

In this case, the CC can correct errors in the first codeword frame if, at most, t errors have occurred in the first blocklength. Such a CC is called a t-error-per-blocklength-correcting CC or, more simply, a t-error-correcting CC.

Another distance between codewords of a CC which is commonly used is the free distance, d_{free}:

$$d_{free} = \max_{\ell} d_\ell .$$

Since, clearly,

$$d = d_{m+1} \leq d_{m+2} \leq \cdots \leq d_{free} ,$$

designing a CC with minimum distance d guarantees that the code has a free distance of d or greater. Note that at least $L = m + 1$ codeword frames are required to compute d_{free}.

Associated with the free distance d_{free} is the free length n_{free}. The

free length of a CC is the length of the non-zero segment of a smallest, non-zero, weight codeword. Hence, $d = d_{free}$ if $m + 1 = n_{\infty}$, and $d < d_{free}$ if $m + 1 < n_{free}$. For a number of useful CCs, $n_{free} = m + 1$, so, for many CCs, $d = d_{free}$, the minimum distance actually equals the free distance.

To avoid catastrophic error propagation, assume G(D) in Eq. (4) to be a basic encoder [3]. The Smith normal form of a basic encoder [3] is

$$G(D) = A(D) \left[I_k, 0 \right] B(D) ,$$

where A(D) and B(D) are, respectively, $k \times k$ and $n \times n$ invertible matrices over F[D] and I_k is a $k \times k$ identity matrix.

Let matrix B[D] in the above Smith normal form be partitioned as

$$B(D) = \left[B_1(D)^T, B_2(D)^T \right]^T ,$$

where $B_1(D)$ consists of the first k rows of B(D) and "T" denotes matrix transpose. Similarly, let

$$B(D)^{-1} = \left[\overline{B}_1(D), \overline{B}_2(D) \right] ,$$

where $\overline{B}_1(D)$ consists of the first k columns of $B(D)^{-1}$. Since $B(D) \cdot B(D)^{-1} = I_n$, the following identities evidently hold:

$$\begin{aligned} B_1(D) \cdot \overline{B}_1(D) &= I_k \quad , \quad B_1(D) \cdot \overline{B}_2(D) = 0 \\ B_2(D) \cdot \overline{B}_1(D) &= 0 \quad , \quad B_2(D) \cdot \overline{B}_2(D) = I_{n-k} . \end{aligned} \tag{6}$$

A parity-check matrix H(D) is an $(n - k) \times n$ matrix of rank $(n - k)$, satisfying

$$G(D) \cdot H^T(D) = 0 . \tag{7}$$

From Eqs. (6) and (7) it is seen next that

$$H(D) = \overline{B}_2(D)^T \tag{8}$$

has the properties of a parity-check matrix H(D) associated with G(D).

By Eq. (3), the CC generated by G(D) is the set

$$C = \left\{ \underline{y}(D) = \left[y_1(D), \ldots, y_n(D) \right] \;\middle|\; \underline{y}(D) = \underline{x}(D)\, G(D) \right\}. \tag{9}$$

It is now shown also that

$$C = \left\{ \underline{y}(D) = \left[y_1(D), \ldots, y_n(D) \right] \;\middle|\; \underline{y}(D)\, H^T(D) = 0 \right\}, \tag{10}$$

where H(D) is given in Eq. (8). To see this, denote the right side of Eq. (10) by C_H. Clearly, an element of set C, as given in Eq. (9), belongs to C_H, and hence $C \subseteq C_H$.

Next, suppose $\underline{y}_1(D)$ is an element of set C_H, i.e., by Eqs. (8) and (10),

$$\underline{y}_1(D)\, H^T(D) = \underline{y}_1(D)\, \overline{B}_2(D) = 0 .$$

But, by definition, $\overline{B}_2(D)$ consists of the last (n - k) columns of $B(D)^{-1}$, so that

$$\overline{B}_2(D) = B(D)^{-1} \begin{bmatrix} 0 \\ I_{n-k} \end{bmatrix}, \tag{11}$$

where "0" denotes a block of k rows of zeros and I_{n-k} is the (n - k) row identity matrix. Thus, $y_1(D)$ satisfies the equation

$$\underline{y}_1(D)\, B^{-1}(D) \begin{bmatrix} 0 \\ I_{n-k} \end{bmatrix} = 0 .$$

The most general solution of this equation for $\underline{y}_1(D)\, B^{-1}(D)$ is

$$\underline{y}_1(D)\, B^{-1}(D) = \left[\tau_1(D), \ldots, \tau_k(D), 0, \ldots, 0\right] = \left[\underline{\tau}(D), 0\right] ,$$

where $\tau_j(D)$ for $1 \leq j \leq k$ can be chosen to be any arbitrary element of F[D]. Solving for $\underline{y}_1(D)$ yields, finally, by Eq. (5),

$$\underline{y}_1(D) = \underline{\tau}(D) \left[I_k, 0\right] B(D) = \underline{\tau}(D)\, A^{-1}(D)\, G(D) ,$$

which belongs to set C, as given in Eq. (9). Thus, $C_H \subseteq C$ and Eq. (10) is proved.

The fact that the CC given by set C in Eq. (9) can be characterized by Eq. (10) is used in the following section to find the coset of solutions to the syndrome equation. This coset is used then as a basis for algebraic syndrome decoding.

II. METHOD OF ALGEBRAIC SYNDROME DECODING

In this section, the syndrome equation is defined and the general solution of the syndrome equation is shown to be a coset of the convolutional code. From the syndrome, a system of linear equations is obtained for the j-th message vector $\underline{v}_j$ and a possible error vector $\underline{e}_j$ in terms of the m previously computed message vectors $\underline{v}_{j-1}, \ldots, \underline{v}_{j-m}$, and $\underline{z}_j$, the j-th received vector. The problem of algebraic syndrome decoding is to solve this set of equations recursively for each j under the constraint that t errors occur per blocklength. This method of decoding is quite different from syndrome feed-back decoding.

Let $\underline{y}(D)$ in Eq. (3) be transmitted and $\underline{z}(D)$ be received. Then,

$$\underline{z}(D) = \underline{y}(D) + \underline{e}(D) , \tag{12}$$

where $\underline{e}(D)$ is the D-transform of the error sequence. By Eqs. (12) and (7),

the syndrome of the received sequence is

$$\begin{aligned} \underline{s}(D) &= \underline{z}(D) \cdot H^T(D) = \left[\underline{y}(D) + \underline{e}(D)\right] \cdot H^T(D) \\ &= \underline{e}(D) \cdot H^T(D) \ . \end{aligned} \tag{13}$$

The problem of syndrome decoding is, given $\underline{s}(D) = \underline{z}(D) \cdot H^T(D)$, to solve the syndrome equation

$$\underline{s}(D) = \underline{z}(D)\, H^T(D) = \underline{e}(D)\, H^T(D) \ , \tag{14a}$$

or its equivalent,

$$\left(\underline{e}(D) - \underline{z}(D)\right) H^T(D) = 0 \ , \tag{14b}$$

for all solutions $\underline{e}(D)$.

By Eqs. (10) and (9), the term $\left(\underline{e}(D) - \underline{z}(D)\right)$ in Eq. (14b) must be some code sequence $\underline{v}(D)\, G(D)$. Hence, the most general solution of the syndrome equation, Eq. (14a), is

$$\underline{e}(D) = \underline{z}(D) + \underline{v}(D)\, G(D) \ , \tag{15}$$

where $\underline{v}(D)$ is the D-transform of an arbitrary message-like sequence $\underline{v} = \left[\underline{v}_0, \ldots, \underline{v}_j, \ldots\right]$ of k-vectors $\underline{v}_j \in V_k(F)$.

Equation (15) shows that the most general solution of the syndrome equation, Eq. (14a), for $\underline{e}(D)$ is the error coset

$$C_z = \left\{ \underline{e}(D) = \underline{z}(D) + v(D)\, G(D) \;\middle|\; \underline{v}(D) = \left[v_1(D), \ldots, v_k(D)\right] \right\} \tag{16}$$

of code C, defined by either Eq. (9) or Eq. (10). A minimization of the Hamming weights over all elements of coset C_z yields the standard minimum-error estimate $\hat{x}(D)$ for message $\underline{x}(D)$. Efficient methods for achieving this minimization include the Viterbi algorithm and all sequential decoding

methods for convolutional codes.

Another syndrome decoding method, different from the above and from feed-back decoding, to find $\hat{x}(D)$ is to utilize decoding methods for CCs which are similar to the basic algebraic techniques of block codes. Thus, if the CC is capable of correcting t-errors per blocklength, then, under the assumption that t errors actually occur, it is desired to solve Eq. (15) algebraically for $\underline{v}(D)$ in a manner which yields for estimate $\hat{x}(D)$ the original message sequence x(D). To accomplish this, first find the so-called state-vector equations for the solution of the syndrome equation, Eq. (15).

By Eq. (5), an equating of coefficients on both sides of Eq. (15) yields

$$\underline{e}_j = \underline{z}_j + \sum_{i=0}^{m} \underline{v}_{j-i} \, G_i \ , \tag{17a}$$

where, initially,

$$\underline{v}_{-1} = \underline{v}_{-2} = \ldots = \underline{v}_{-m} = 0 \tag{17b}$$

as the state-vector equations associated with all elements of the error coset Eq. (16). The goal of algebraic syndrome decoding of a convolutional code is to fill in the details of the following algorithm:

Frame 0: Given Eq. (17b) and the constraint that no more than t errors occurred in $\underline{e}_0, \underline{e}_1, \ldots, \underline{e}_m$, solve Eq. (17a) for $\underline{v}_0$.

Frame 1: Given Eq. (17b), solution $\underline{v}_0$ and the constraint that no more than t errors occurred in $\underline{e}_1, \underline{e}_2, \ldots, \underline{e}_m$,

solve Eq. (17a) for $\underline{v}_1$.

. . .
. . .
. . .

Frame m: Given Eq. (17b), solutions $\underline{v}_0, \underline{v}_1, \ldots, \underline{v}_{m-1}$ and the constraint that no more than t errors occurred in $\underline{e}_m$, $\underline{e}_{m+1}, \ldots, \underline{e}_{2m}$, solve Eq. (17a) for $\underline{v}_n$. Use solutions $\underline{v}_0, \underline{v}_1, \ldots, \underline{v}_m$ to construct estimate $\hat{x}_0$ of original message.

. . .
. . .
. . .

Frame j: Given Eq. (17b), solutions $\underline{v}_{j-m}, \underline{v}_{j-m+1}, \ldots, \underline{v}_{j-1}$ and the constraint that no more the t errors occurred in $\underline{e}_j$, $\underline{e}_{j+1}, \ldots, \underline{e}_{j+m}$, solve Eq. (17a) for $\underline{v}_j$. Use solutions $\underline{v}_{j-m}, \underline{v}_{j-m+1}, \ldots, \underline{v}_{j+1}$ to construct estimate $\hat{x}_0$ of original message.

Since each symbol $\hat{x}_j$ is obtained algebraically under the assumption that no more than

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor$$

errors actually occur, the estimated symbols $\hat{x}_j$ must equal the original transmitted symbols x_j for $j = 0, 1, \ldots$. This type of decoding algorithm will be demonstrated by example in the next section, using a systematic dual-k code.

III. ALGEBRAIC SYNDROME DECODING OF SYSTEMATIC DUAL-k CONVOLUTIONAL CODES

Systematic dual-k convolutional codes are defined to be (n, 1) CCs of rate 1/n, of memory m = 1, and with symbols in the finite or Galois field $G(2^k)$. See Odenwalder's paper [4] for a description and definition of the original non-systematic dual-k convolutional code.

The generating matrix G(D) of the systematic dual-k CC in the form of Eq. (4) is

$$G(D) = G_0 + G_1 D \ , \tag{18a}$$

where

$$G_0 = [1, 1, \ldots, 1] \text{ and}$$
$$G_1 = \left[0, g_2, g_3, \ldots, g_n\right] , \tag{18b}$$

where $g_j \in GF(2^k)$ and $g_1 \neq 0$ for (j = 2, ..., n). That G(D) is systematic is seen from its form, i.e.,

$$G(D) = \left[1, 1 + g_2 D, \ldots, 1 + g_n D\right] . \tag{19}$$

From Eq. (18) or (19) and the different definitions of distance given in Section I, it is readily seen that the minimum distance d equals the free distance and that

$$d = d_{free} = 2 n - 1 \ .$$

Hence, if no more than t symbols occur in the first two codeword frames, one has

$$2 t + 1 \leq d = 2 n - 1 \text{ or}$$
$$t \leq n - 1 \ , \tag{20}$$

so that the maximum number of errors that can be corrected per blocklength is $t = n - 1$. In other words, the systematic dual-k CCs of rate $1/n$ are $t = n - 1$ symbol-errors-per-blocklength, correcting, convolutional codes.

Example of Algebraic Syndrome Decoding. Let generating matrix for example be

$$G(D) = [1, 1 + D] \tag{21}$$

over $GF(2^k)$. This has form of generating matrix in Eq. (7) so that $n = 2$ and $G(D)$ is generating matrix of a (2, 1) systematic dual-k convolutional code of memory $m = 1$. By Eq. (20), this code is one symbol-error correcting-per-blocklength, where blocklength is $m + 1 = 2$.

A substitution of Eq. (21) into Eq. (15) yields

$$\sum_{i=0}^{\infty} \left[e_{i1}, e_{i2} \right] D^i = \sum_{i=0}^{\infty} \left[v_i, v_i + v_{i-1} \right] D^i + \sum_{i=0}^{\infty} \left[z_{1i}, z_{2i} \right] D^i \quad .$$

Next, equating of coefficients obtains

$$e_{1i} = v_i + z_{1i} \quad \text{and} \quad e_{2i} = v_i + v_{i-1} + z_{2i} \tag{22}$$

for $(i = 0, 1, \ldots)$, where initially $v_{-1} = 0$.

The solution of the two equations in Eq. (22) for v_{i-1} and v_i is

$$v_{i-1} = z_{1i} + z_{2i} + e_{1i} + e_{2i} \tag{I}$$

$$v_i = z_{1i} \qquad + e_{1i}. \tag{II}$$

It is desired now to obtain a recursive estimate $\hat{x}_{i-1}$ of the transmitted message from (I) and (II) on the assumption that no more than one error occurs per blocklength of 2. To accomplish this, the following two lemmas are needed:

Lemma 1. Given Eqs. (I) and (II), if no errors in the i-th frame, i.e., $\underline{e}_i = \left[e_{1i}, e_{2i} \right] = 0$, then

$$v_{i-1} = z_{1i} + z_{2i} \qquad \text{(III)}$$

$$v_i = z_{1}i \ . \qquad \text{(IV)}$$

Proof: A substitution of $\left[e_{1i}, e_{2i}\right] = [0, 0]$ into (I) and (II) yields (III) and (IV) directly. Hence, lemma is true.

Lemma 2. Given (I) and (II), then

$$v_{i-1} = z_{1i} + z_{2i} \quad \text{iff} \quad \left[e_{1i}, e_{2i}\right] = [0, 0] \ ,$$

where "iff" denotes "if and only if."

Proof: If $\left[e_{1i}, e_{2i}\right] = [0, 0]$, then by (I), $v_{i-1} = z_{1i} + z_{2i}$. Conversely if $v_{i-1} = z_{1i} + z_{2i}$, then, by (I),

$$e_{1i} + e_{2i} = 0 \ .$$

But, since $0 \leq t \leq 1$, $(e_{1i} \neq 0)$ and $(e_{2i} \neq 0)$ can not both be true. Thus, $\underline{e}_i$ must be zero and lemma is true.

By Lemma 1, if no error occurs at frame i, then v_{i-1} can be uniquely determined so that the best estimate of the original message at frame i-1 is $\hat{x}_{i-1} = v_{i-1}$. Also, by Lemma 2, one can determine whether or not an error has occurred at frame i by testing whether or not the previously estimated message $v_{i-1} = x_{i-1}$ equals $z_{1i} + z_{2i}$.

In order to solve (I) and (II) frame-by-frame on the assumption that no more than one error occurs every two frames, define two auxiliary variables:

i) Let A denote the delayed and correct message symbol; and

ii) Let ε denote a binary variable such that

$(\varepsilon = 0)$ iff (previous frame had no error),

i.e.,

$$(\varepsilon = 0) \text{ iff } (\underline{e}_{i-1} = 0) .$$

iii) Next, define the binary variable λ at frame i in the following manner:

$$(\lambda = 1) \text{ iff } \left(z_{1i} + z_{2i} = A_i\right) .$$

Finally, let "x ← y" denote operation, "variable x is replaced by y."

In terms of variables A, ε, and λ, defined above, $\hat{x}_{i-1}$ is found frame-by-frame at frame i, utilizing Lemmas 1 and 2, as follows:

Frame, i = -1: $\underline{e}_{-1} = 0$ and $\underline{v}_{-1} = 0$. Hence, initially, $\varepsilon \leftarrow 0$ and $A \leftarrow 0$.

Frame, i = 0: Since $\varepsilon = 0$, $\underline{e}_{-1} = 0$ so that, by (IV), $A = v_{-1} = 0$, the initial condition for v_i. By Lemma 2,

(a) If $z_{10} + z_{20} = 0 = A$, i.e., if $\lambda = 1$, then no error occurred at frame 0 and $\varepsilon \leftarrow 0$, i.e., ε is set to zero. But also, by Lemma 1, $v_0 = z_{10} = \hat{x}_0$, so that $A \leftarrow v_0 = z_{10}$.

(b) Or, if $z_{10} + z_{20} \neq 0 = A$, i.e., if $\lambda = 1$, then at least one error occurred at frame 0 and $\varepsilon \leftarrow 1$. However, since only one error is allowed for one blocklength of 2 frames, one concludes that no error can occur at the next frame, i.e., at i = 1 or frame 1,

so that, by Lemma 1, the data $[z_{11}, z_{21}]$ from frame 1 is used to compute $v_0 = z_{11} + z_{21} = \hat{x}_0$, the estimated message for frame 0, and $v_1 = z_{11} = \hat{x}_1$, the estimated message for frame 1.

Frame, i = 1: A. If $\varepsilon = 0 \left(\underline{e}_0 = 0\right)$, then, by Lemma 2,

(a) If $z_{11} + z_{21} = A$, i.e., if $\lambda = 1$, then no error occurred at frame 1 and $\varepsilon \leftarrow 0$. Thus, by Lemma 1, $v_0 = z_{11} + z_{21} = \hat{x}_0$ and $v_1 = z_n = \hat{x}_1$, and $A \leftarrow v_1 = z_{11}$.

(b) Or, if $z_{11} + z_{21} \neq A$, i.e., if $\lambda = 0$, then no error occurred at frame 1, and $\varepsilon \leftarrow 1$. Since only one error is allowed per blocklength, must have at <u>next</u> frame, i.e,, at i = 2, by Lemma 1, $v_1 = z_{12} + z_{22} = \hat{x}_1$ and $v_2 = z_{12} = \hat{x}_2$.

B. On the other hand, if $\varepsilon = 1 \left(\underline{e}_0 \neq 0\right)$, then since only one error is allowed per blocklength, must have $\underline{e}_1 = 0$, and $\varepsilon \leftarrow 0$. Also, by Lemma 1, must have

$v_0 = z_{11} + z_{21} = \hat{x}_0$ and $v_1 = z_{11}$ so that $A \leftarrow v_1 = z_{11}$.

. . .

Frame i = j: A. If $\varepsilon = 0 \left(\underline{e}_{j-1} = 0\right)$, then, since only one error

(a) If $z_{1j} + z_{2j} = A$, i.e., if $\lambda = 1$, then no error occurred at frame j and $\varepsilon \leftarrow 0$. Thus, by Lemma 1, $v_{j-1} = z_{1j} + z_{1j} = \hat{x}_{j-1}$ and $v_j = z_{1j} = \hat{x}_j$ and $A \leftarrow v_j = z_{1j}$.

(b) Or, if $z_{ij} + z_{2j} \neq A$, i.e., if $\lambda = 0$, then at least one error occurred at frame j, and $\varepsilon \leftarrow 1$. Since only

one error is allowed per blocklength, must have at next frame, i.e., at $i = j$, by Lemma 1, $v_j = z_{1,\,j+1} + z_{2,\,j+1} = \hat{x}_j$ and $v_{j+1} = z_{1,\,j+1} = \hat{x}_j$.

B. On the other hand, if $\varepsilon = 1 \left(\underline{e}_{j-1} \neq 0 \right)$, then, since only one error is allowed per blocklength, must have $\underline{e}_j = 0$, and $\varepsilon \leftarrow 0$. Also, by Lemma 1, must have $v_{j-1} = z_{1j} + z_{2j} = \hat{x}_{j-1}$ and $v_j = z_{1j}$ so that $A \leftarrow v_j = z_{1j}$.

In Fig. 1, the above decoding algorithm is presented in flow chart form, along with the delayed output equation,

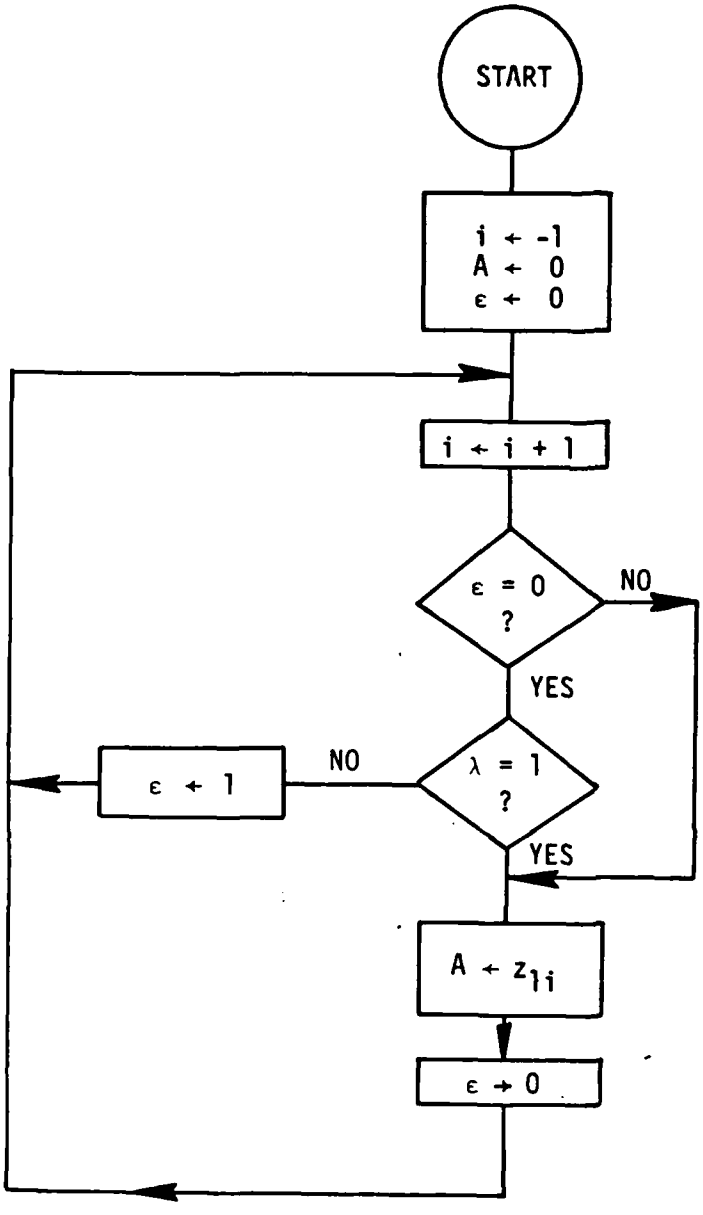
$$\hat{x}_{i-1} = \left(\lambda_i \vee \varepsilon_i \right) \left(z_{1i} + z_{2i} \right) + \bar{\varepsilon}_i \, \bar{\lambda}_i \, A_i \, . \tag{23}$$

Since it is assumed that only one error occurred per blocklength, the estimated or corrected message sequence, $\hat{x}_{i-1}$ in Eq. (23), actually equals the original message sequence, x_{i-1}, delayed by one frame. The symbol "v" denotes logical "or" and the bar over the variable $\bar{\varepsilon}$ denotes negation, i.e., $\bar{\varepsilon} = 1 - \varepsilon$. As an output, Eq. (23) is read as follows:

$$\text{If } \left[\left(\lambda_i = 1 \right) \text{ or } \left(\varepsilon_i = 1 \right) \right], \; \hat{x}_{i-1} = \left(z_{1i} + z_{2i} \right), \text{ or}$$

$$\text{if } \left[\left(\lambda_i = 0 \right) \text{ and } \left(\varepsilon_i = 0 \right) \right], \; \hat{x}_{i-1} = A_i \, .$$

It is not difficult to demonstrate by simple cases that the algorithm in Fig. 1 for the above example will compute the original message sequence x_{i-1} as long as no more than one error occurs per blocklength. Work is continuing on the development of a similar algebraic syndrome decoding algorithm



Delayed Message Output $\hat{x}_{i-1}$:

$$\hat{x}_{i-1} = (\lambda_i \vee \varepsilon_i)(z_{1i} + z_{2i}) + \bar{\varepsilon}_i \bar{\lambda}_i A_i$$

Definitions: A = delayed corrected message symbol
($\varepsilon = 0$) iff (previous frame was error free)
($\lambda = 1$) iff $(z_{1i} + z_{2i} = A_i)$

Fig. 1 -- Flow chart of decoding algorithm with delayed message output $\hat{x}_{i-1}$

for both the systematic and non-systematic (n, 1) dual-k convolutional code. Recently an example of such an algorithm for correcting two symbol errors per blocklength, the n = 2 case, was completed. This result plus others on the more general cases will be reported on elsewhere.

REFERENCES

1. Reed, I. S., *Pruned Error-Trellis Decoding of Certain Non-Systematic Convolutional Codes*, First Quarterly Report submitted to the Office of Naval Research on Contract N00014-84-C-0720 by Adaptive Sensors, Inc., December 1984.

2. Blahut, R. E., *Theory and Practice of Error Control Codes*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

3. Forney, G. D., "Convolutional Codes: Algebraic Structure," *IEEE Transactions on Information Theory*, IT-16, 1970, pp. 720-738.